Vue.js 框架单元测试

首先声明一点，长期以来，前端开发的单元测试并不是在前端的开发过程中所必须的，也不是每个前端开发工程师所注意和重视的，甚至很多前端开发人员的开发过程中根本就没有单元测试的概念。但是随着前端的发展,每个项目的复杂化、代码组件化要求、前端项目优化和重构需求，前端工程中的单元测试流程就显得很有其必要。

一.什么是单元测试

首先要明确测试是什么：为检测特定的目标是否符合标准而采用专用的工具或者方法进行验证，并最终得出特定的结果。

对于前端开发过程来说，这里的特定目标就是指我们写的代码，而工具就是我们需要用到的测试框架(库)、测试用例等。检测出的结果就是展示测试是否通过或者给出测试报告，这样才能方便问题的排查和后期的修正。

二.为什么要单元测试

2.1 单元测试的意义

对于现在的前端工程，一个标准完整的项目，测试是非常有必要的。很多时候我们只是完成了项目而忽略了项目测试的部分，测试的意义主要在于下面几点：

■ 测试驱动开发被证明是有效的软件编写原则，它能覆盖更多的功能接口。

■ 快速反馈你的功能输出，验证你的想法。

■ 保证代码重构的安全性，没有一成不变的代码，测试用例能给你多变的代码结构一个定心丸。

■ 易于测试的代码，说明是一个好的设计。做单元测试之前，肯定要实例化一个东西，假如这个东西有很多依赖的话，这个测试构造过程将会非常耗时，会影响你的测试效率，怎么办呢？要依赖分离，一个类尽量保证功能单一，比如视图与功能分离，这样的话，你的代码也便于维护和理解。

2.2 为什么需要前端单元测试

■ 首先是一个前端单元测试的根本性原由：JavaScript 是动态语言，缺少类型检查，编译期间无法定位到错误；JavaScript 宿主的兼容性问题。比如 DOM 操作在不同浏览器上的表现。

■ 正确性：测试可以验证代码的正确性，在上线前做到心里有底。

■ 自动化：当然手工也可以测试，通过 console 可以打印出内部信息，但是这是一次性的事情，下次测试还需要从头来过，效率不能得到保证。通过编写测试用例，可以做到一次编写，多次运行。

■ 解释性：测试用例用于测试接口、模块的重要性，那么在测试用例中就会涉及如何使用这些 API。其他开发人员如果要使用这些 API，那阅读测试用例是一种很好地途径，有时比文档说明更清晰。

■ 驱动开发，指导设计：代码被测试的前提是代码本身的可测试性，那么要保证代码的可测试性，就需要在开发中注意 API 的设计，TDD 将测试前移就是起到这么一个作用。

■ 保证重构：互联网行业产品迭代速度很快，迭代后必然存在代码重构的过程，那怎么才能保证重构后代码的质量呢？有测试用例做后盾，就可以大胆的进行重构

三 . 怎么进行单元测试

3.1 原则

■ 测试代码时，只考虑测试，不考虑内部实现

■ 数据尽量模拟现实，越靠近现实越好

■ 充分考虑数据的边界条件

■ 对重点、复杂、核心代码，重点测试

■ 利用 AOP(beforeEach、afterEach)，减少测试代码数量，避免无用功能

■ 测试、功能开发相结合，有利于设计和代码重构

3.2 两个常用的单元测试方法论 (https://www.jianshu.com/p/aa1cb52cc0a5)

在单元测试中，常用的方法论有两个：TDD（测试驱动开发）&& BDD（行为驱动开发）

四.基于 Vue.js 的单元测试

4.1.测试环境

Vue 官方有一个 Vue Test utils 的测试工具，推荐一个叫做 karma 的自动化测试，它产生一个 Web 服务环境来运行项目代码，并且执行测试，该工具在 Vue 中的主要作用是将项目运行在各种主流 Web 浏览器进行测试。换句话说，它是一个测试工具，能让你的代码在浏览器环境下测试。以下 DEMO 的测试运行器基于 Karma。

相关环境配置：

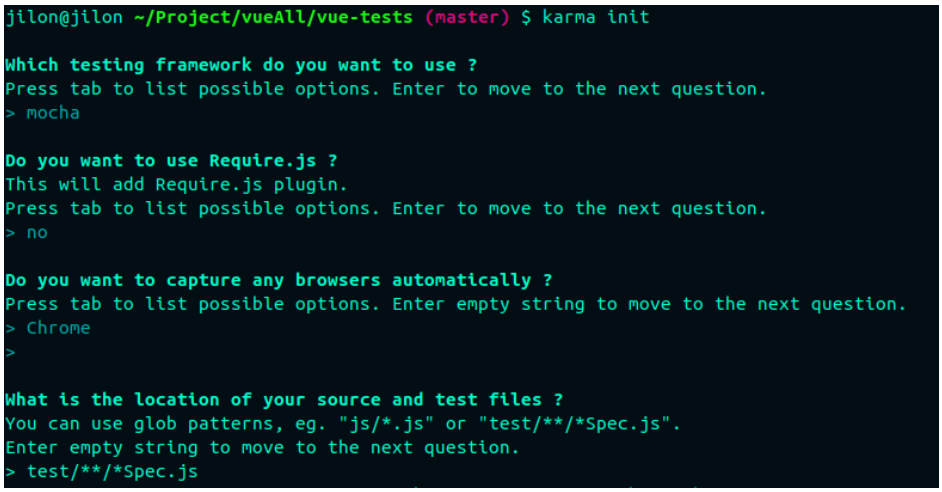
nvm 管理 node 版本　node@8.11.1
npm@5.6.0
webpack@2.3.3

通过 npm 安装一下依赖包：

- 官方测试工具库 ：
  @vue/test-utils@1.0.0-beta.10
  https://vue-test-utils.vuejs.org/zh/
- 测试运行器
  karma@2.0.0
  官网：http://karma-runner.github.io/latest/index.html
- JS 测试框架
  mocha@3.2.0
  官网：https://mochajs.org/
- 断言库
  chai@3.5.0
  官网：https://www.chaijs.com/

4.2.项目配置

1、终端执行 karma init，主目录生成 karma.conf.js



注意的有：选择测试框架 mocha，选择浏览器 Chrome，输入测试文件路径

2、 修改 karma.conf.js 文件配置，主要的配置有一下：

浏览器：

```
browsers: ['Chrome'],
```

框架和库，工作路径相关：

```
frameworks: ['mocha', 'sinon-chai'],
```

读取文件路径（根据项目中实际路径配置）：

```
files: [
  'test/unit/**/*.spec.js'
],
```

文件预处理配置：

```
preprocessors: {
  'test/unit/**/*.spec.js': ['webpack', 'sourcemap']
},
```

浏览器运行：

```
singleRun: false,
```

3、 webpack 测试环境配置

karma.conf.js 文件中引入 webpack.test.conf.js，config.set 中配置 webpack。

```
const webpackConfig = require('./build/webpack.test.conf.js')
```

```
webpack: webpackConfig,
```

4、 配置 package.json

```
"scripts": {
  "dev": "node build/dev-server.js",
  "start": "node build/dev-server.js",
  "build": "node build/build.js",
  "unit": "cross-env BABEL_ENV=test karma start karma.conf.js",
  "e2e": "node test/e2e/runner.js",
  "test": "npm run unit"
},
```

5、 运行 npm run test 或者 npm run unit，可以查看单元测试结果。

4.3.DEMO

运行 npm run test

```
jilon@jilon ~/Project/vueAll/vue-tests (master) $ npm run test

> vue-tests@1.0.0 test /home/jilon/Project/vueAll/vue-tests
> npm run unit


> vue-tests@1.0.0 unit /home/jilon/Project/vueAll/vue-tests
> cross-env BABEL_ENV=test karma start karma.conf.js

Hash: 9acd869ca4806c0c16ad
Version: webpack 2.7.0
Time: 38ms
webpack: Compiled successfully.
webpack: Compiling...
webpack: wait until bundle finished:
Hash: ca733d818e890a97ba0a
Version: webpack 2.7.0
Time: 413ms
```

显示每个单元测试文件的编译执行过程。

```
                         Asset     Size  Chunks                    Chunk Names
test/unit/specs/Hello.spec.js  971 kB       0  [emitted]  [big]  test/unit/specs/Hello.sp
ec.js
 test/unit/specs/List.spec.js    1 MB       1  [emitted]  [big]  test/unit/specs/List.spe
c.js
chunk    {0} test/unit/specs/Hello.spec.js (test/unit/specs/Hello.spec.js) 358 kB [entry]
    [0] (webpack)/buildin/global.js 509 bytes {0} {1} [built]
    [1] ./src/components/Hello.vue 1.61 kB {0} [built]
    [2] ./~/vue/dist/vue.esm.js 325 kB {0} {1} [built]
    [3] ./~/babel-loader/lib!./~/vue-loader/lib/selector.js?type=script&index=0!./src/com
ponents/Hello.vue 2.76 kB {0} [built]
    [4] ./test/unit/specs/Hello.spec.js 565 bytes {0} [built]
    [5] ./~/css-loader?{"minimize":false,"sourceMap":false}!./~/vue-loader/lib/style-comp
iler?{"id":"data-v-15dbd5a7","scoped":true,"hasInlineConfig":false}!./~/vue-loader/lib/se
lector.js?type=styles&index=0!./src/components/Hello.vue 417 bytes {0} [built]
    [6] ./~/css-loader/lib/css-base.js 2.26 kB {0} [built]
    [7] ./~/process/browser.js 5.42 kB {0} {1} [built]
    [8] ./~/setimmediate/setImmediate.js 6.47 kB {0} {1} [built]
    [9] ./~/timers-browserify/main.js 2.02 kB {0} {1} [built]
   [10] ./~/vue-loader/lib/component-normalizer.js 1.27 kB {0} {1} [built]
   [11] ./~/vue-loader/lib/template-compiler?{"id":"data-v-15dbd5a7"}!./~/vue-loader/lib/
selector.js?type=template&index=0!./src/components/Hello.vue 2.13 kB {0} [built]
   [12] ./~/vue-style-loader!./~/css-loader?{"minimize":false,"sourceMap":false}!./~/vue-
loader/lib/style-compiler?{"id":"data-v-15dbd5a7","scoped":true,"hasInlineConfig":false}!
./~/vue-loader/lib/selector.js?type=styles&index=0!./src/components/Hello.vue 1.66 kB {0}
   [13] ./~/vue-style-loader/lib/addStylesClient.js 6.05 kB {0} [built]
   [14] ./~/vue-style-loader/lib/listToStyles.js 639 bytes {0} [built]
chunk    {1} test/unit/specs/List.spec.js (test/unit/specs/List.spec.js) 368 kB [entry] [
rendered]
    [0] (webpack)/buildin/global.js 509 bytes {0} {1} [built]
    [2] ./~/vue/dist/vue.esm.js 325 kB {0} {1} [built]
    [8] ./~/setimmediate/setImmediate.js 6.47 kB {0} {1} [built]
    [9] ./~/timers-browserify/main.js 2.02 kB {0} {1} [built]
   [10] ./~/vue-loader/lib/component-normalizer.js 1.27 kB {0} {1} [built]
   [15] ./~/avoriaz/dist/VueWrapper.js 1.87 kB {1} [built]
   [16] ./~/avoriaz/index.js 136 bytes {1} [built]
```

编译完成后显示测试结果同时打开浏览器。

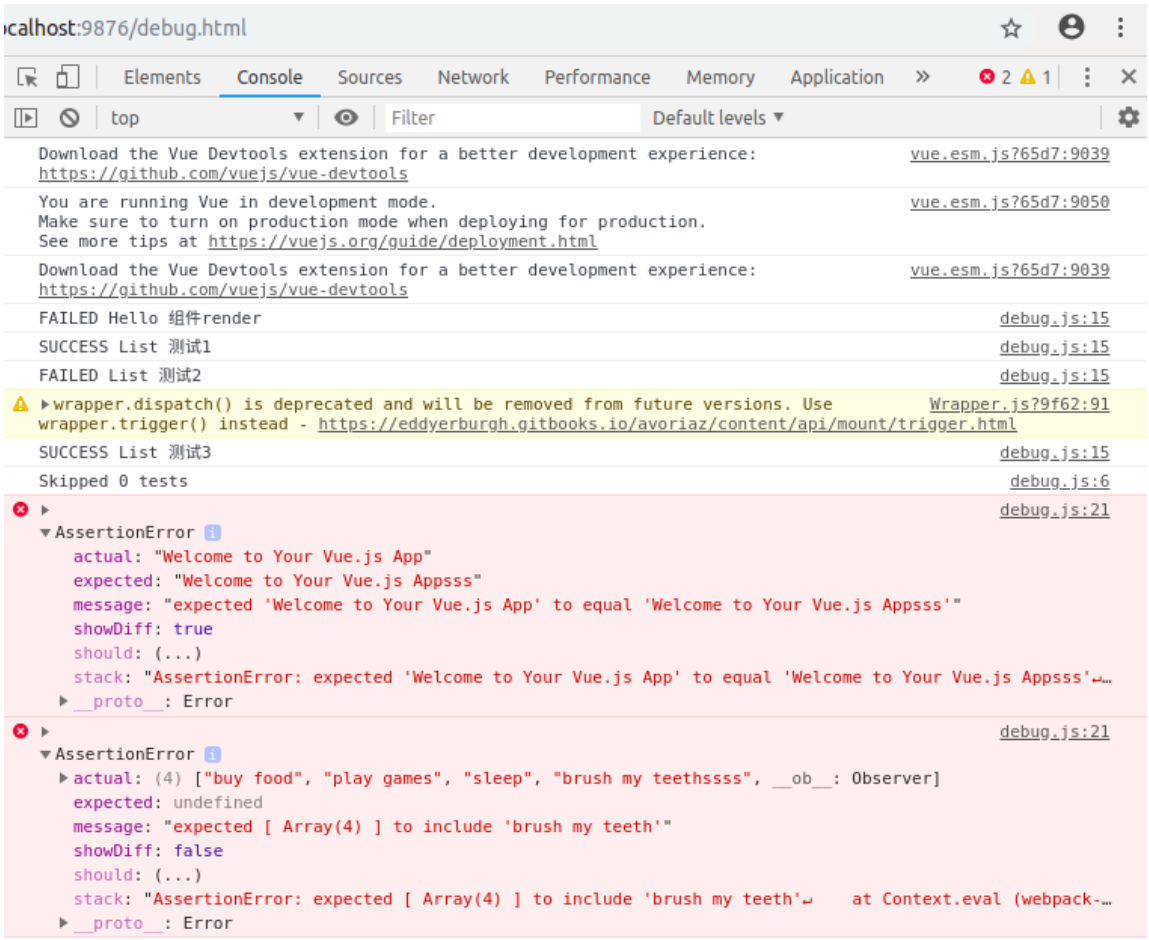
```
webpack: Compiled successfully.
17 03 2019 21:18:33.404:WARN [karma]: No captured browser, open http://localhost:9876/
17 03 2019 21:18:33.412:INFO [karma]: Karma v2.0.0 server started at http://0.0.0.0:9876/
17 03 2019 21:18:33.413:INFO [launcher]: Launching browser Chrome with unlimited concurre
ncy
17 03 2019 21:18:33.416:INFO [launcher]: Starting browser Chrome
17 03 2019 21:18:37.644:INFO [Chrome 70.0.3538 (Linux 0.0.0)]: Connected on socket EFYlKT
IHp8SnLTRKAAAA with id 43916291
INFO LOG: 'Download the Vue Devtools extension for a better development experience:
https://github.com/vuejs/vue-devtools'
INFO LOG: 'You are running Vue in development mode.
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html'
INFO LOG: 'Download the Vue Devtools extension for a better development experience:
https://github.com/vuejs/vue-devtools'

  Hello
    ✗ 组件render
        AssertionError: expected 'Welcome to Your Vue.js App' to equal 'Welcome to Your V
ue.js Appsss'
            at Context.eval (webpack-internal:///4:17:62)


  List
    ✓ 测试1
    ✗ 测试2
        AssertionError: expected [ Array(4) ] to include 'brush my teeth'
            at Context.eval (webpack-internal:///27:36:40)

WARN LOG: 'wrapper.dispatch() is deprecated and will be removed from future versions. Use
 wrapper.trigger() instead - https://eddyerburgh.gitbooks.io/avoriaz/content/api/mount/tr
igger.html'
    ✓ 测试3

Chrome 70.0.3538 (Linux 0.0.0): Executed 4 of 4 (2 FAILED) (0.004 secs / 0.019 secs)
TOTAL: 2 FAILED, 2 SUCCESS
```

显示测试覆盖率

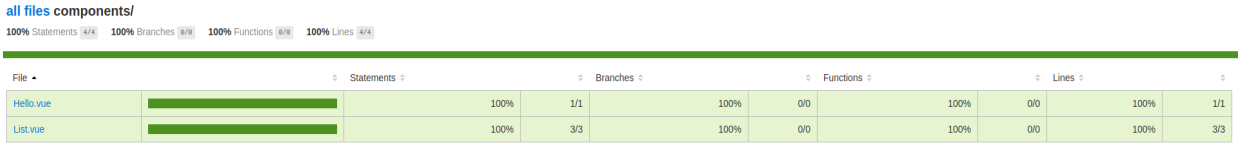
```
=============================== Coverage summary ===============================
Statements   : 100% ( 4/4 )
Branches     : 100% ( 0/0 )
Functions    : 100% ( 0/0 )
Lines        : 100% ( 4/4 )
================================================================================
```

浏览器运行 karma server，地址 localhost:9876，显示运行环境。

点击 DEBUG，DEBUG 页面打开开发者工具，切换到 console 面板，查看测试详情。



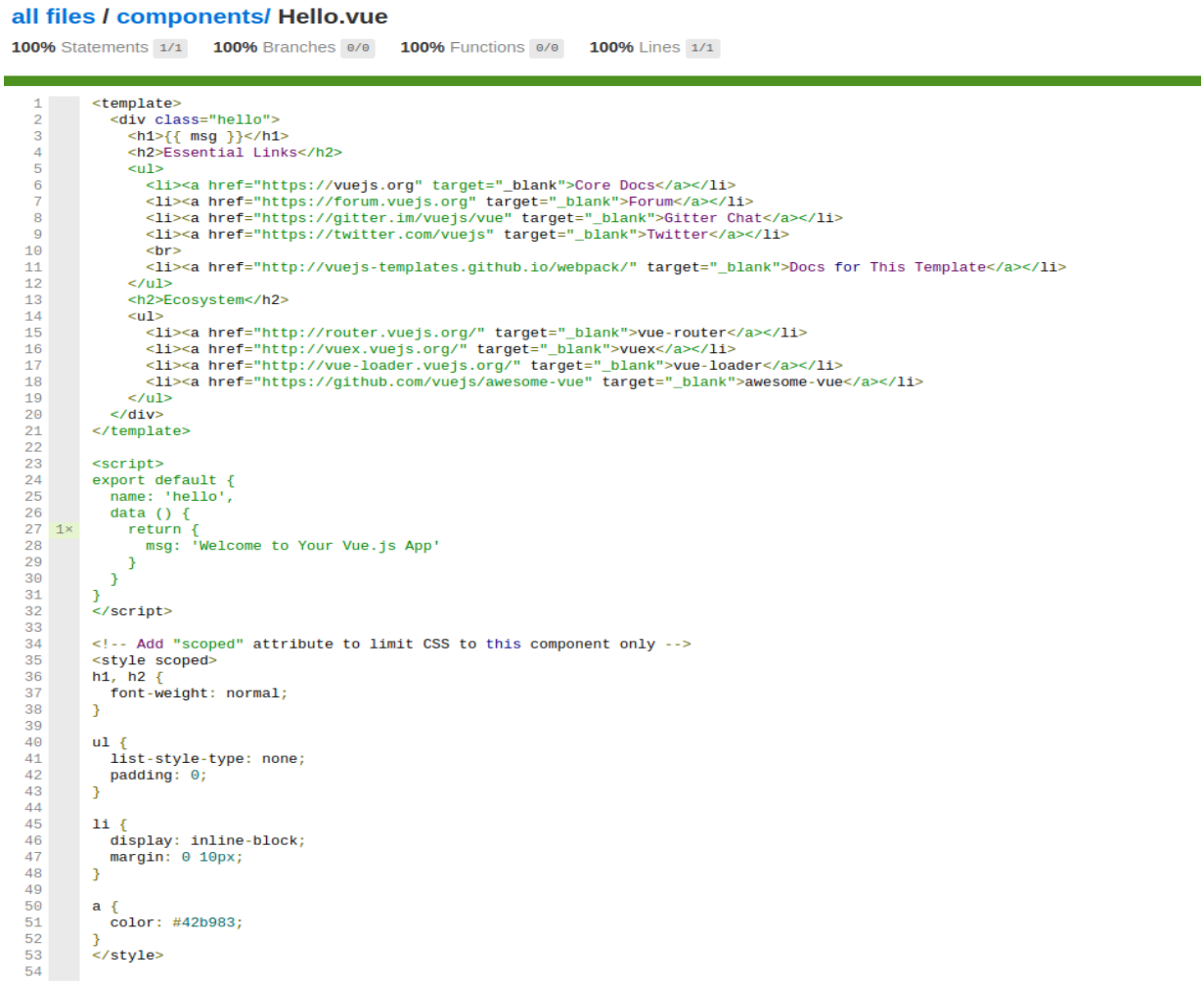
最后会在配置的目录生成测试报告文件，方便回头查看。



all files components/

100% Statements 4/4 100% Branches 0/0 100% Functions 0/0 100% Lines 4/4

File		Statements		Branches		Functions		Lines	
Hello.vue		100%	1/1	100%	0/0	100%	0/0	100%	1/1
List.vue		100%	3/3	100%	0/0	100%	0/0	100%	3/3



all files / components/ Hello.vue

100% Statements 1/1 100% Branches 0/0 100% Functions 0/0 100% Lines 1/1

```
1  <template>
2    <div class="hello">
3      <h1>{{ msg }}</h1>
4      <h2>Essential Links</h2>
5      <ul>
6        <li><a href="https://vuejs.org" target="_blank">Core Docs</a></li>
7        <li><a href="https://forum.vuejs.org" target="_blank">Forum</a></li>
8        <li><a href="https://gitter.im/vuejs/vue" target="_blank">Gitter Chat</a></li>
9        <li><a href="https://twitter.com/vuejs" target="_blank">Twitter</a></li>
10       <br>
11       <li><a href="http://vuejs-templates.github.io/webpack/" target="_blank">Docs for This Template</a></li>
12     </ul>
13     <h2>Ecosystem</h2>
14     <ul>
15       <li><a href="http://router.vuejs.org/" target="_blank">vue-router</a></li>
16       <li><a href="http://vuex.vuejs.org/" target="_blank">vuex</a></li>
17       <li><a href="http://vue-loader.vuejs.org/" target="_blank">vue-loader</a></li>
18       <li><a href="https://github.com/vuejs/awesome-vue" target="_blank">awesome-vue</a></li>
19     </ul>
20   </div>
21 </template>
22
23 <script>
24 export default {
25   name: 'hello',
26   data () {
27 1×  return {
28       msg: 'Welcome to Your Vue.js App'
29     }
30   }
31 }
32 </script>
33
34 <!-- Add "scoped" attribute to limit CSS to this component only -->
35 <style scoped>
36 h1, h2 {
37   font-weight: normal;
38 }
39
40 ul {
41   list-style-type: none;
42   padding: 0;
43 }
44
45 li {
46   display: inline-block;
47   margin: 0 10px;
48 }
49
50 a {
51   color: #42b983;
52 }
53 </style>
54
```

五.单元测试用例

Vue Test Utils 通过将它们隔离挂载，然后模拟必要的输入 (prop、注入和用户事件) 和对输出 (渲染结果、触发的自定义事件) 的断言来测试 Vue 组件。我们可以模拟测试鼠标键盘事件，用户交互，异步行为，并且配合 Vue Router 使用，当然可以在组件中测试 Vuex。

关于测试断言的语法，具体参考官网 API。https://www.chaijs.com/

DEMO 中用到的两个简单的测试用例：

```
describe('Hello', () => {
  it('组件render', () => {
    const Constructor = Vue.extend(Hello)
    const vm = new Constructor().$mount()
    expect(vm.$el.querySelector('.hello h1').textContent).to.equal('Welcome to Your Vue.js Appsss')
  })
})
```

```
describe('List', () => {
  it('测试1', () => {
    const Constructor = Vue.extend(List);
    const ListComponent = new Constructor().$mount();
    expect(ListComponent.$el.textContent).to.contain('play gamess');
  })
  it('测试2', () => {
    const Constructor = Vue.extend(List);
    const ListComponent = new Constructor().$mount();
    ListComponent.newItem = 'brush my teethssss';
    // simulate click event
    const button = ListComponent.$el.querySelector('button');
    const clickEvent = new window.Event('click');
    button.dispatchEvent(clickEvent);
    ListComponent._watcher.run();
    // assert list contains new item
    expect(ListComponent.$el.textContent).to.contain('brush my teeth');
    expect(ListComponent.listItems).to.contain('brush my teeth');
  })
  it('测试3', () => {
    const ListComponent = mount(List);
    ListComponent.setData({
      newItem: 'brush my teeth',
    });
    const button = ListComponent.find('button')[0];
    button.dispatch('click');
    expect(ListComponent.text()).to.contain('brush my teeth');
    expect(ListComponent.data().listItems).to.contain('brush my teeth');
  })
})
```